IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.:	**:**	**10/671,056**	**Confirmation No. 5833**
Applicant:	**:**	**Brokenshire, et al.**	
Filed:	**:**	**September 25, 2003**	
Group Art Unit:	**:**	**2192**	
Examiner:	**:**	**Wei, Zheng**	
Docket No.	**:**	**AUS920030701US1**	
Customer No.	**:**	**40412**	

§

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Mailing or Transmission
I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 or electronically transmitted to the U.S. Patent and Trademark Office on the date shown below.

/Leslie A. Van Leeuwen, Reg. No. 42,196/	Jan. 9, 2008
Leslie A. Van Leeuwen, Reg. No. 42,196	Date

RESPONSE TO NOTICE OF NON-COMPLIANT APPEAL BRIEF

Sir:

A. INTRODUCTORY COMMENTS

In response to the notice of non-compliant Appeal Brief having a mailing date of December 27, 2007, with a one-month statutory period for response set to expire on January 27, 2008, Applicants respectfully submit this Supplemental Appeal Brief. As required by the Notice, Applicants have included an Evidence Appendix and a Related Proceedings Appendix in this Supplemental Appeal Brief. Other than these additions, there are no changes to the Appeal Brief as originally filed on December 12, 2007.

No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and the undersigned hereby authorizes the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

B. REAL PARTY IN INTEREST

The real party in interest in this appeal is International Business Machines Corporation, which is the assignee of the entire right, title, and interest in the above-identified patent application.

C. RELATED APPEALS AND INTERFERENCES

With respect to other prior or pending appeals, interferences, or judicial proceedings that are related to, will directly affect, be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such prior or pending appeals, interferences, or judicial proceeding known to Appellants, Appellants' legal representative, or assignee.

D. STATUS OF CLAIMS

1. Total number of claims in application

There are 17 claims pending. Three claims are independent claims (1, 8, and 14), and the remaining claims are dependent claims.

2. Status of all claims in application

- Claims canceled: 2, 9, and 15
- Claims withdrawn from consideration but not canceled: none
- Claims pending: 1, 3-8, 10-14, and 16-20
- Claims allowed: None
- Claims rejected: 1, 3-8, 10-14, and 16-20

3. Claims on appeal

Claims 1, 3-8, 10-14, and 16-20 are on appeal.

E. STATUS OF AMENDMENTS

All amendments have been entered in this case. No amendments have been made to the claims after the Final Office Action.

F. SUMMARY OF CLAIMED SUBJECT MATTER

Appellants provide a concise summary of the claimed subject matter as follows. Claims 1, 8, and 14 are independent claims. Note that claims 1 and 3-7 are method claims, claims 8 and 10-13 are information handling system claims, and claims 14 and 16-20 are computer program product claims. An information handling system capable of implementing Appellants' invention, as claimed in independent claim 8, is shown in Figure 51, and described in Appellants' specification on page 62, line 1 through page 64, line 19. Support for independent computer program product claim 14 is described in Appellants' specification on page 64, line 20 through page 65, line 7. In addition, support for each of the method steps and logic elements limitations of the independent claims are discussed below. The specific citations to Appellants' Figures and Specification are meant to be exemplary in nature, and do not limit the scope of the claims. In particular, the citations below do not limit the scope of equivalents as provided under 35 U.S.C. § 112, sixth paragraph.

As claimed in independent claim 1, Appellants claim a method for compiling source code for a plurality of heterogeneous processor types by receiving source code that includes a plurality of source code subtasks (See Figure 43, reference numerals 4300-4310; page 44, line 27 through page 45, line 4); independently selecting a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19), the independent selection comprising: selecting a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19); and selecting

a second processor type from the plurality of heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19); and creating an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type (See Figure 43, reference numerals 4330-4370; page 45, line 3 through page 46, line 17).

As claimed in independent claim 8, Appellants claim an information handling system (See Figure 51, reference numeral 5105; page 62, line 1 through page 64, line 19), a plurality of heterogeneous processors (See Figure 51, reference numerals 5110, 5145, 5165, and 5185; page 62, lines 1 through 15); a memory accessible by the heterogeneous processors (See Figure 51, reference numeral 5115; page 62, lines 26-27); one or more nonvolatile storage devices accessible by the heterogeneous processors (See Figure 51, reference numeral 5120, 5159, 5179, and 5199; page 62, line 26 through page 63, line 10); and a source code compilation tool for compiling source code (See Figure 43, reference numeral 4320, page 45, lines 3 through 4), the source code compilation tool comprising software effective to receive source code that includes a plurality of source code subtasks (See Figure 43, reference numerals 4300-4310; page 44, line 27 through page 45, line 4); independently select a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19), comprising: select a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19); and select a second processor type from the plurality of

heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19); and create an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type (See Figure 43, reference numerals 4330-4370; page 45, line 3 through page 46, line 17).

As claimed in independent claim 14, Appellants claim a computer program product (See page 64, line 20 through page 65, line 7) stored on a computer operable media (See Figure 51, reference numeral 5120, 5159, 5179, and 5199; page 62, line 26 through page 63, line 10; page 64, line 20 through page 65, line 7), the computer operable media containing instructions for execution by a computer (See pages page 64, line 20 through page 65, line 7), which, when executed by the computer, cause the computer to implement a method for compiling source code for a plurality of heterogeneous processors by receiving source code that includes a plurality of source code subtasks (See Figure 43, reference numerals 4300-4310; page 44, line 27 through page 45, line 4); independently selecting a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19), the independent selection comprising: selecting a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19); and selecting a second processor type from the plurality of heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type (See Figure 47, reference numerals 4710-4720, 4740-4750, and 4780-

4790; page 53, lines 1-15; page 54, lines 6-15; and page 55, lines 10-19); and creating an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type (See Figure 43, reference numerals 4330-4370; page 45, line 3 through page 46, line 17).

G. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1, 3-8, 10-14, and 16-20 on appeal stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Oram (Oram et al., Managing Projects with Make) in view of Stallman (Richard M Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1).

H. ARGUMENT

1. Claims 1, 3-8, 10-14, and 16-20 Are Patentable over Oram in view of Stallman.

Independent claim 1 claims a method for compiling source code for a plurality of heterogeneous processor types with limitations of:

1) receiving source code that includes a plurality of source code subtasks;

2) independently selecting a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks, the independent selection comprising:

3) selecting a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code; and

4) selecting a second processor type from the plurality of heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type; and

5) creating an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type.

Appellants individually retrieve source code subtasks included in a source code file, select a particular processor type for each subtask, and create processor-type specific object code for each of the subtasks, which results in a single object file that includes multiple object code subtasks. In contrast, Oram discloses the use of C programming "make" commands to selectively compile source code into an object code file. Oram's make command, however, is designed to produce an object file for **each** source code file. Oram states:

> "To compile the statements you want, define the proper symbols in CFLAGS. For instance, the following *make* command compiles ***all*** source files with the BSD and STATS symbols defined:" (page 79, emphasis added)

As can be seen from the above excerpt, Oram's "make" command generates a single object file from source code. The Office Action points to an excerpt in Oram to reject Appellants' last limitation but, after further review of Oram, Oram does not teach such limitation. On page 79, Oram states:

> "all you have to do is make full_test, and make passes the right –D option to each command as follows:
>
> $ make full_test
> make trac.o "CFFLAGS = -DSTATS –DBSD"
> cc –DSTATS DBSD –c trac.c
> make main.o "CFLAGS = -DBSD"
> cc –DBSD –c main.c
> cc –o full_test trac.o main.o"

The Office Action uses the excerpt's last line "*cc –o full_test trac.o main.o*" to reject claim 1's last limitation of "*creating an **object file** that includes a first object code…and a second object code.*" However, Oram's last line includes a "-o" syntax and, therefore, does not create an object file, but rather creates an **executable** file

named "full_test" using two separately generated object files "trac.o" and "main.o." As known by those skilled in the art, object files are created using a "-c" syntax. Therefore, Oram never teaches or suggests "*creating* ***an object file*** *that includes* ***a first object code*** *corresponding to the first source code subtask and* ***a second object code*** *corresponding to the second source code subtask, wherein the* ***first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type***" as claimed by Appellants. The Office action does not suggest that Stallman teaches claim 1's last limitation and, indeed, Stallman does not teach such limitation.

Based on the foregoing, Appellants respectfully submit that the rejection of Appellants' independent claim 1 over Oram in view of Stallman has been overcome and, therefore, claim 1 is allowable over Oram in view of Stallman. Claim 8 is an information handling claim including similar limitations as claim 1 and, therefore, is allowable over Oram in view of Stallman for at least the same reasons that claim 1 is allowable as discussed above. Claim 14 is computer program product claim including similar limitations as claim 1 and, therefore, is allowable over Oram in view of Stallman for at least the same reasons that claim 1 is allowable as discussed above.

Each of the remaining claims 3-7, 10-13, and 16-20 depend, either directly or indirectly, upon one of the allowable independent claims 1, 8 or 14. Therefore, each of claims 3-7, 10-13, and 16-20 are allowable for at least the same reasons as there respective independent claims are allowable as discussed above.

Conclusion

For the foregoing reasons, Appellants respectfully submit that claims 1, 3-8, 10-14, and 16-20 are patentable, and, accordingly, Appellants respectfully request that the Examiner's claim rejections be reversed and claims 1, 3-8, 10-14, and 16-20 be allowed.

Respectfully submitted,

By /Leslie A. Van Leeuwen, Reg. No. 42,196/

Leslie A. Van Leeuwen, Reg. No. 42,196
Van Leeuwen & Van Leeuwen
Attorney for Applicants
Telephone: (512) 301-6738
Facsimile: (512) 301-6742

I. CLAIMS APPENDIX

1. A method for compiling source code for a plurality of heterogeneous processor types, said method comprising:

receiving source code that includes a plurality of source code subtasks;

independently selecting a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks, the independent selection comprising:

selecting a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code; and

selecting a second processor type from the plurality of heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type; and

creating an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type.

2. (Canceled)

3. The method as described in claim 1 wherein the selection of the first processor type is performed during compilation, the method further comprising:

retrieving the first source code subtask from the plurality of source code subtasks;

determining whether the first source code subtask includes a program directive corresponding to one of the plurality of processors; and

performing the selection of the first processor type in response to the determination.

4. The method as described in claim 1 further comprising:

retrieving the first source code subtask from the plurality of source code subtasks; and

compiling the first source code subtask, the compiling resulting in byte code.

5. The method as described in claim 4 further comprising:

sending the byte code to a client over a computer network, wherein the byte code is adapted to be translated into client-specific object code by the client whereby the client-specific object code is formatted based upon a processor type that is located at the client.

6. The method as described in claim 1 further comprising:

retrieving the first source code subtask from the plurality of source code subtasks;

identifying one or more operations included in the first source code subtask;

matching one or more of the operations with one of the processor types from the plurality of heterogeneous processor types; and

performing the selection of the first processor type in response to the matching.

7. The method as described in claim 1 further comprising:

receiving a processor-specific command, the processor specific command identifying a processor type from the plurality of heterogeneous processor types; and

performing the selection of the first processor type based upon the processor-specific command.

8. An information handling system comprising:

a plurality of heterogeneous processors;

a memory accessible by the heterogeneous processors;

one or more nonvolatile storage devices accessible by the heterogeneous processors; and

a source code compilation tool for compiling source code, the source code compilation tool comprising software code effective to:

receive source code that includes a plurality of source code subtasks from one of the nonvolatile storage devices;

independently select a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks comprising:

select a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code; and

select a second processor type from the plurality of heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type; and

create an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type.

9. (Canceled)

10. The information handling system as described in claim 8 wherein the selection of the first processor type is performed during compilation, wherein the software code is further effective to:

retrieve the first source code subtask from the plurality of source code subtasks located in one of the nonvolatile storage devices;

determine whether the first source code subtask includes a program directive corresponding to one of the plurality of processors; and

perform the selection of the first processor type in response to the determination.

11. The information handling system as described in claim 8 wherein the software code is further effective to:

retrieve the first source code subtask from the plurality of source code subtasks; and

compile the first source code subtask, the compiling resulting in byte code.

12. The information handling system as described in claim 11 wherein the software code is further effective to:

send the byte code to a client over a computer network, wherein the byte code is adapted to be translated into client-specific object code by the client whereby the client-specific object code is formatted based upon a processor type that is located at the client.

13. The information handling system as described in claim 8 wherein the software code is further effective to:

retrieve the first source code subtask from the plurality of source code subtasks located in one of the nonvolatile storage devices;

identify one or more operations included in the first source code subtask;

match one or more of the operations with one of the processor types from the plurality of heterogeneous processor types; and

perform the selection of the first processor type in response to the matching.

14. A computer program product stored on a computer operable media, the computer operable media containing instructions for execution by a computer, which, when executed by the computer, cause the computer to implement a method for compiling source code for a plurality of heterogeneous processors, the method comprising:

receiving source code that includes a plurality of source code subtasks;

independently selecting a processor type from the plurality of heterogeneous processor types for each of the plurality of source code subtasks, the independent selection comprising:

selecting a first processor type from the plurality of heterogeneous processor types for a first source code subtask included in the source code; and

selecting a second processor type from the plurality of heterogeneous processor types for a second source code subtask included in the source code, wherein the second processor type is different than the first processor type; and

creating an object file that includes a first object code corresponding to the first source code subtask and a second object code corresponding to the second source code subtask, wherein the first object code is adapted to be processed by the first processor type and the second object code is adapted to be processed by the second processor type.

15. (Canceled)

16. The computer program product as described in claim 14 wherein the selection of the first processor type is performed during compilation, the method further comprising

retrieving the first source code subtask from the plurality of source code subtasks;

determining whether the first source code subtask includes a program directive corresponding to one of the plurality of processors; and

performing the selection of the first processor type in response to the determination.

17. The computer program product as described in claim 14 wherein the method further comprises :

retrieving the first source code subtask from the plurality of source code subtasks; and

compiling the first source code subtask, the compiling resulting in byte code.

18. The computer program product as described in claim 17 wherein the method further comprises :

sending the byte code to a client over a computer network, wherein the byte code is adapted to be translated into client-specific object code by the client whereby the client-specific object code is formatted based upon a processor type that is located at the client.

19. The computer program product as described in claim 14 wherein the method further comprises :

retrieving the first source code subtask from the plurality of source code subtasks;

identifying one or more operations included in the first source code subtask;

matching one or more of the operations with one of the processor types from the plurality of heterogeneous processor types; and

performing the selection of the first processor type in response to the matching.

20. The computer program product as described in claim 14 wherein the method further comprises :

receiving a processor-specific command, the processor specific command identifying a processor type from the plurality of heterogeneous processor types; and

performing the selection of the first processor type based upon the processor-specific command.

J. EVIDENCE APPENDIX

Not applicable.

K. RELATED PROCEEDINGS APPENDIX

Not applicable.